Hindawi
International Journal of Aerospace Engineering
Volume 2017, Article ID 6143727, 9 pages
https://doi.org/10.1155/2017/6143727

Research Article

Architecture Level Safety Analyses for Safety-Critical Systems

K. S. Kushal, Manju Nanda, and J. Jayanthi

Aerospace Electronics & Systems Division, CSIR-National Aerospace Laboratories, Bangalore, Karnataka, India

Correspondence should be addressed to K. S. Kushal; ksk261188@gmail.com

Received 24 August 2016; Revised 23 November 2016; Accepted 15 December 2016; Published 15 January 2017

Academic Editor: Paul Williams

The dependency of complex embedded Safety-Critical Systems across Avionics and Aerospace domains on their underlying software and hardware components has gradually increased with progression in time. Such application domain systems are developed based on a complex integrated architecture, which is modular in nature. Engineering practices assured with system safety standards to manage the failure, faulty, and unsafe operational conditions are very much necessary. System safety analyses involve the analysis of complex software architecture of the system, a major aspect in leading to fatal consequences in the behaviour of Safety-Critical Systems, and provide high reliability and dependability factors during their development. In this paper, we propose an architecture fault modeling and the safety analyses approach that will aid in identifying and eliminating the design flaws. The formal foundations of SAE Architecture Analysis & Design Language (AADL) augmented with the Error Model Annex (EMV) are discussed. The fault propagation, failure behaviour, and the composite behaviour of the design flaws/failures are considered for architecture safety analysis. The illustration of the proposed approach is validated by implementing the Speed Control Unit of Power-Boat Autopilot (PBA) system. The Error Model Annex (EMV) is guided with the pattern of consideration and inclusion of probable failure scenarios and propagation of fault conditions in the Speed Control Unit of Power-Boat Autopilot (PBA). This helps in validating the system architecture with the detection of the error event in the model and its impact in the operational environment. This also provides an insight of the certification impact that these exceptional conditions pose at various criticality levels and design assurance levels and its implications in verifying and validating the designs.

1. Introduction

Systematic analyses of the architectural models modeled using the Model-Based Engineering (MBE) [1] practices, early and at every abstraction level, imbibe a greater confidence in the integration of the system. The creation and analysis of architectural models of a system support prediction and understanding of the system's capabilities and its operational quality attributes. These attributes include performance, reliability, reusability, safety, and security. All along the developmental lifecycle, the faults such as their failure modes and their propagation effects, at system-level, can be predicted. Such issues remain unnoticed until system integration and testing. This proves to be a costly rework resulting in an unaccounted project time, cost, and maintenance.

For safety-critical advanced complex embedded systems, the system design and development are in compliance with the safety standards and engineered with practices as specified by MIL-STD882 [2], SAE ARP-4761 [3], and DO-178B/C [4]. The process of development, management, and controlling these systems in conformance with the safety practices proves to have an impact on the system requirements, postsystem integration, and test. With the evolution of the system, availability and reliability of these models are to be consistent and this poses a great challenge.

These safety practices include various availability and reliability prognosis with the help of system architectural models. Model-Based Engineering approaches for safety analyses address these issues and prove to provide consolidated information about the informal requirements and the architecture model of the system. The safety analyses performed on a system also take into consideration the physical environment of its deployment and functioning. Due to insufficient support of the formal languages trend is to make use of architecture description languages such as Architecture Analysis & Design Language (AADL) and Society of Automotive Engineers (SAE) standard. AADL, a high-level architectural descriptive language, basically provides a platform for overall

integration of various system recommended components via formal semantics and syntax. This component-based modeling language is extended with the introduction of sublanguages as Annexes. AADL is packaged with multiple Annex sublanguages such as Error Model Annex (EAnnex) and Behaviour Annex (BAnnex) as standards. The EAnnex standard is suitably augmented with safety semantics and ontology of fault propagation, supporting error annotations on the architectural models [5]. This thus enables the component error models and their interactions to be considered in context to the system architecture modeled using AADL.

This paper presents our contributions as a case study implementation (Speed Control Unit of Power-Boat Autopilot) to the standard approach for the illustration of its application. The paper is organized as follows. Firstly, we summarize the concept of Architecture Analysis & Design Language (SAE AADL) and Error Model Annex (EAnnex/EMV2). Next we provide an illustration of the architecture fault model specification for Speed Control Unit of a Power-Boat Autopilot (PBA). We also discuss the various safety analyses methods involved in MIL-STD882 safety practice. Finally, we conclude the paper with the assessment of these safety analyses based on the architecture fault models.

2. Error Model Annex in Architecture Analysis & Design Language (AADL)

Architecture Analysis & Design Language (AADL), an SAE International standard, is a unified framework providing extensive formal foundations for Model-Based Engineering (MBE) practices. These practices extend throughout the system design, integration, and assurance with safety standards. AADL distinctly represents a system hardware and software components and their interactions via interfaces. Critical real-time computational factors such as performance, dependability, safety, security, and data integrity can be rigorously analysed with AADL.

AADL also integrates custom analyses and specification techniques during the engineering process. This allows in the development and analysis of a single, unified system architectural model. AADL can be extended using the specialized language constructs that can be attached to the components of the architectural model defined by AADL. These components are reinforced with additional characteristics and requirements, referred to as Annex languages. The architectural model components are annotated with these properties and Annex language clauses for functional and nonfunctional analyses. Error Model Annex (EMV), which is an extension of AADL, aids in describing the failure conditions and fault propagations as error events, propagations, occurrence, and their distribution properties. With the integration of these constructs in the AADL model/s, as shown in Figure 1 [6], the existing components are extended as current models liable for Safety Evaluation and Analyses. This can be done with the help of the algorithms in OSATE or by using other third party tools.

(i) Error Annex. The Error Model Annex (EAnnex) is a sublanguage of AADL. This sublanguage extension includes

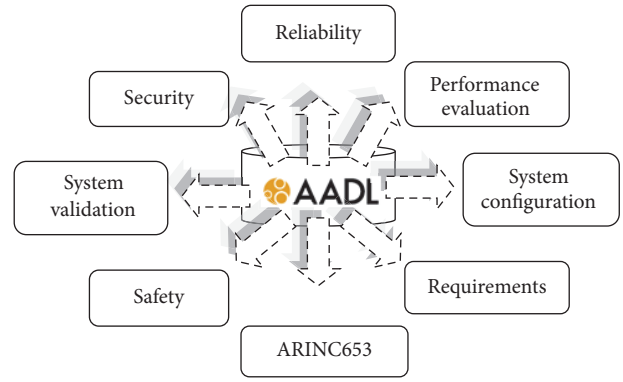

Figure 1: AADL ecosystem.

the analyses of the runtime architectures. The EAnnex [7, 8] annotates the hardware and the software component architectures with error states, error events, error transitions, and error propagations that may affect the component interacting with each other. In Error Model Annex subclause conditions can be specified under which the errors are propagated through designated component ports. Error Model Annex basically helps in defining the fault models, hazards, fault propagation, failure modes, and effects, as well as specifying compositional fault behaviour. AADL Error Model Annex supports architectural fault modeling at three levels of abstraction [9].

(1) Modeling of faults in systems and their implications on other dependent components of the physical environment of its operation through propagation of these faults (including Hazard identification, fault impact analysis)

(2) Modeling of faults occurring in a component of the system and analysing the behaviour of the same across various modes termed as failure modes and their effects on other components and their related propagations and being also inclusive of the recovery strategies involved

(3) Compositional abstraction of system error behaviour in terms of its subsystems

Error Model Annex (EMV2) overlays major focus on the standards set of error types and error propagation, defined by AADL as a standard syntactic construct through the introduction of Annex libraries. These Annex libraries provide an overlook of the formally specified error propagation behaviours [10, 11]. Some of the common error types are as follows [9].

(1) Commission and Omission Errors. They represent loss of message/command and failure to provide readings from a component.

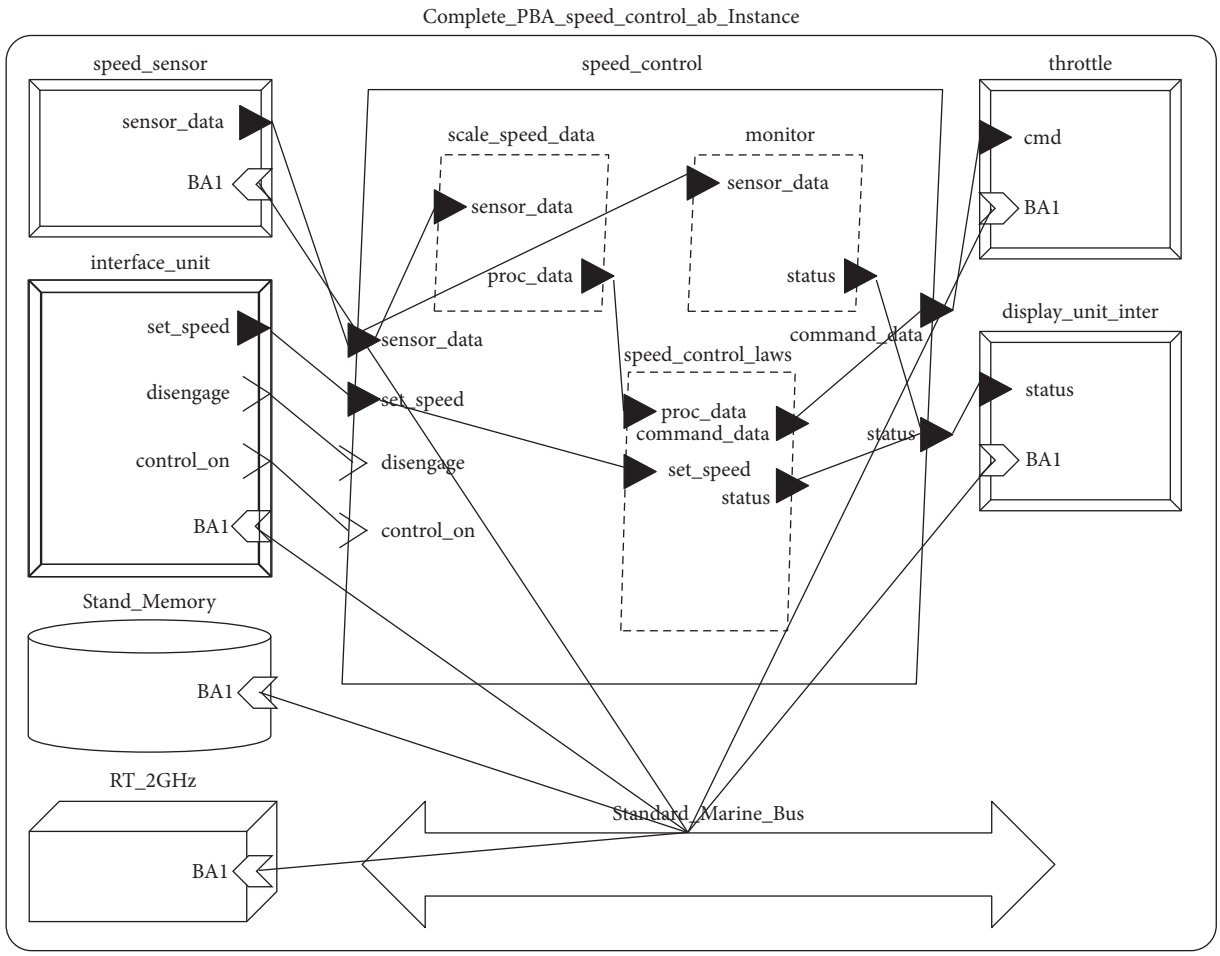

Figure 2: PBA Speed Control Unit without error specification.

(2) Timing Errors. They represent arrival rate, service too early or late, and unsynchronized rate.

(3) Value Errors. They represent individual service item error or errors in a sequence of values.

(4) Replication Errors. They represent replicates of states or services being communicated.

(5) Concurrency Errors. They represent accessing shared logical or physical resources.

Along with these the error model types can be referenced in the Error Model Annex subclause. The constructs for the EMV2 are similar to the syntax and style as defined for AADL. An exception is that any set of textual language constructs can be included within an Annex that includes Object Constraint Language (OCL) [12] or a temporal logic notation [13].

3. Implementation of Proposed Research

In this section we exhibit the architecture fault modeling in AADL, along with the extension of EMV2, at three levels of abstraction with a suitable case study, Speed Control Unit of Power-Boat Autopilot (PBA). This unit is a simplified speed control model, including a pilot interface unit for input of relevant Power-Boat Autopilot information, a speed sensor that sends speed data to the PBA, the PBA controller, a throttle actuator that responds to the commands specified by the PBA controller, and a display unit. The type definitions defining the component, component names, their runtime category, and interfaces are identified and defined. The speed sensor, pilot interface, throttle actuator, and the display unit are modeled as devices, while the PBA control functions are represented as process, as shown in Figure 2. With all these we perform the safety analyses with the specification of the source of error and its propagation across the system and its components. This is carried out by defining the error states and their corresponding compositional fault behaviour. This is followed by the expansion of the fault logic with respect to its error behaviour related to each component of the system and its response to the failures.

(ii) Specification of Error Source and Propagation. The source of errors and their propagation with respect to each component of a system in PBA Speed Control Unit is defined, as shown in Box 1. In the case study `on_flow_src` related

```
device interface
features
set_speed : out data port;
disengage : out event port;
control_on : out event port;
BA1 : requires bus access Marine.Standard;

flows
on_flow_src : flow source set_speed;

annex EMV2{**
use types ErrorModelLibrary;
use behavior ErrorModelLibrary::Simple;

error propagations
set_speed: out propagation{NoValue};
disengage: out propagation{NoService};
control_on: out propagation{NoService};

flows
fPath_Src: error source set_speed{NoValue};
end propagations;
```

Box 1: Error source and propagation.

```
system implementation
Complete.PBA_speed_control_ab
subcomponents
speed_sensor : device sensor.speed;
throttle : device actuator.speed;
interface_unit : device interface.pilot;
speed_control : process control_ex.speed;
display_unit_inter : device display_unit;
RT_2GHz : processor Real_Time.two_GHz;
Standard_Marine_Bus : bus Marine.Standard;
Stand_Memory : memory RAM.Standard;

annex EMV2{**
use types ErrorModelLibrary;
use behavior ErrorModelLibrary::Simple;

composite error behavior
states
[throttle.Failed and
display_unit_inter.Failed]-> Failed;
[display_unit_inter.Failed]-> Operational;
end composite;
```

Box 2: Composite error behaviour.

to the device, pilot interface unit is sourcing the fault. The component error propagations are also defined with the error `NoValue` & `NoService`.

The component error behaviour is also defined for the system components that correlate to the faults that are possible to occur. Here in this system the `NoValue` due to failure passes on from the pilot interface unit to the throttle actuator. The same is being conveyed to the display unit feature `status`. In addition to this fault, there occurs another propagation of error that is `NoService`. This fault results in the Failed state of the system. Here we can observe that the specification is automatically inherited by the instances of each component and their interactive neighbors. The error propagation paths inherent in such system architecture AADL models form a basis, as a need for the representation of Failure Mode and Effect Analysis (FMEA) and Common Cause Analysis (CCA).

(iii) Composite Error Behaviour. The Error Model Annex library is associated with the state machine defined for the system component model using the declaration **`use behaviour`**, as shown in Box 2. This maps the error state behaviour of the subcomponents (both hardware and software components) onto the error states of the system itself. In this case study of Speed Control Unit of PBA, we have two error states defined for each component, that is, `Failure` and `Failed`. But here we have considered only the `Failed` state as the subcomponent error state and the state `Operational` as the recovery state. We can see in this example that the system error behaviour is mapped from the subcomponent behaviours defined as

`[throttle.Failed and display_unit_inter .Failed] -> Failed.`

We assume that the system fails if either of the devices, that is, throttle actuator or the display unit, behaves in the `Failed` state, while it tends to recover from the `Failed` state and remains to be `Operational` even if the display unit fails, as the speed control unit mainly depends on the throttle command in maintaining and controlling the speed of the PBA.

`[display_unit_inter.Failed] -> Operational.`

This provides a scope for redundancy management for fault management capability of the system as well as seek for extensive solutions for reliability and availability analyses through various hierarchical levels of the system architecture. This methodology is not advisable for Markov Chains as the systems tends to grow quickly with their dependencies among various components within a system, as the number of components increases.

(iv) Component Error Behaviour. The modeler will have the flexibility of analysing the possible error behaviour that may correspond to individual components of a system. This also provides an insight into the component internal failures and the divergent factors that may result in failure mode, in turn having an impact on other components. The case study in this paper specifies that there might be multiple failure modes like `Failure` and `Failed`. In `Failed` mode the entire component is assumed to be redundant while in the `Failure` the component is working but having erroneous outputs/output states, as shown in Box 3.

```
device display_unit
features
status : in data port;
BA1 : requires bus access Marine.Standard;
flows
on_flow_snk : flow sink status;

annex EMV2{**
use types ErrorModelLibrary;
use behavior ErrorModelLibrary::Simple;

error propagations
status: in propagation{NoValue};
Flows
fPath_Snk: error sink status{NoValue};
end propagations;

component error behavior
transitions
t0: Operational -[status{NoValue}] -> Failed;
end component;
```

Box 3: Component error behaviour.

The failure modes are represented using the error states with more likely coupled error behaviour of the subsystem/component. The consistency checker associated with the Error Model Annex abstracts the propagation specification to introduce unique and distinctive error types. While the modeling tool associated with the Error Model Annex validates the organization of the component error behaviour along with the propagation specification specific to each of the components in the system architecture, the actual system architecture must include the Safety System component/s that regulates the fault management and aids in safety analyses.

4. Safety Analyses

Safety Analyses involve various analytical processes such as consistency checks, Fault Tree Analysis (FTA), Failure Modes and Effect Analysis (FMEA), Functional Hazard Assessment (FHA), and Common Mode Assessment (CMA) of the architectural model. The architecture model and its associated fault model are designed and developed in Open Source AADL Tool Environment (OSATE) [14]. It is an Eclipse based AADL modeling framework. There is also need to the safety analysis tool such as OpenFTA [15]. An Open Source tool for FTA is integrated into Eclipse environment, to assist in generation of FTA and its relevant documents, while CMA, FMEA, and FHA reports are generated as a built-in feature from OSATE.

(i) Consistency Checks. The consistency checks at the system integration level scan for the consistency in their functionality and the interfaces between various models/components, as shown in "Consistency Report" Section. This thereby strengthens the Virtual integration and analysis of the architecture model of the system. The consistency of various models deals with their integration feasibility while the consistency of the internal components in a model concentrates on the propagation capabilities, redundancies, and so on. With Error Model Annex the concept of consistency across the error models as specified checks for the consistency with respect to the component error behaviour along with the composite error behaviour of the system. It helps in defining the correctness of the error state as per the components specified in the architectural model. This may be proven with the substantial inclusion of Behaviour Annexes (BAnnex) [16] along with the Error Model Annex. The consistency report generated by the OSATE plugin for the case study is as follows.

Consistency Report

> Warning! Complete_PBA_speed_control_ab_Instance: C13: component Complete_PBA_speed_control_ab_Instance does not define occurrence for and state Failed
>
> Complete_PBA_speed_control_ab_Instance:C13: component Complete_PBA_speed_control_ab_Instance has consistent probability values for state Operational
>
> Warning! Complete_PBA_speed_control_ab_Instance: C13: component Complete_PBA_speed_control_ab_Instance does not define occurrence for and state Failed
>
> Warning! Complete_PBA_speed_control_ab_Instance: C13: component Complete_PBA_speed_control_ab_Instance does not define occurrence for and state Failed
>
> Warning! Complete_PBA_speed_control_ab_Instance: C13: component Complete_PBA_speed_control_ab_Instance does not define occurrence for and state Failed
>
> Complete_PBA_speed_control_ab_Instance: C13: component Complete_PBA_speed_control_ab_Instance has consistent probability values for state Failed

(ii) Fault Tree Analysis (FTA). It is a widely used safety and reliability analysis [17] feature in aerospace, medical electronics, and industrial automation industries [18]. In this analysis the major focus is on the top-level event (Minimal Cut-Set), from a set of combinations of basic events (Faults). It provides a hierarchical representation of the errors of the system (top-level event) from the basic events, related to components as specified in component error behaviour, in the form of a tree. OSATE depicts this composite error behaviour of the system from the underlying component error behaviours as a fault tree that represents specific error state of the system. This is achieved in the form of two files from OSATE for the representation of the fault tree, one being the database of primary events (*.ped*), as shown in Figure 4, causing the top-level error event, and the Fault Tree Analysis file (*.fta*). These files are viewed using OpenFTA, as shown in Figure 3.

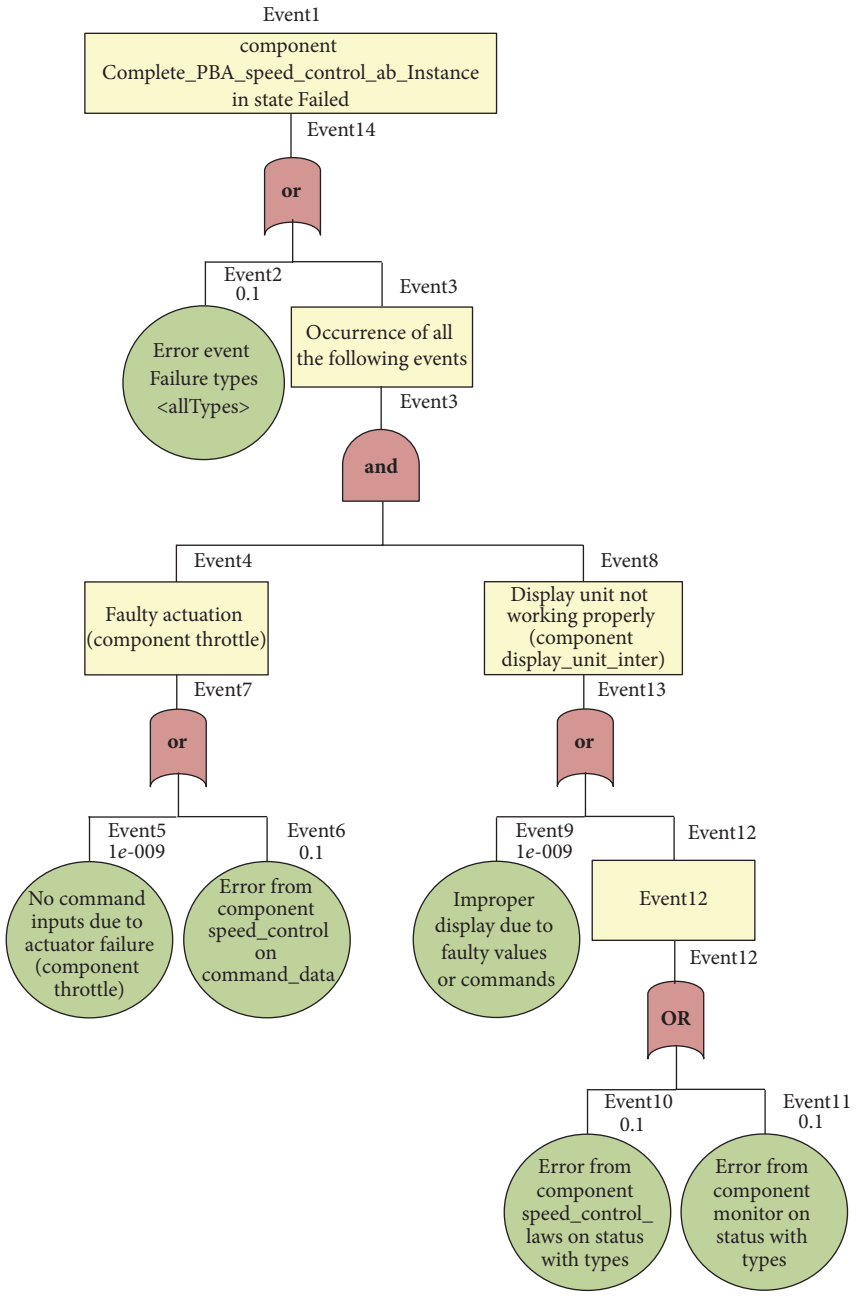

Figure 3: FTA view for PBA Speed Control Unit in OpenFTA.

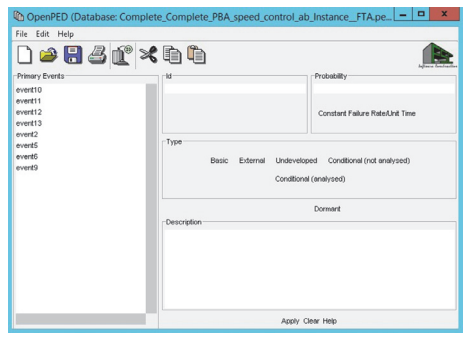

Figure 4: PED view in OpenFTA.

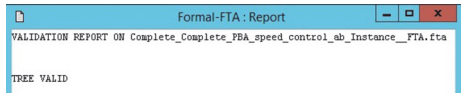

Figure 5: FTA validation report.

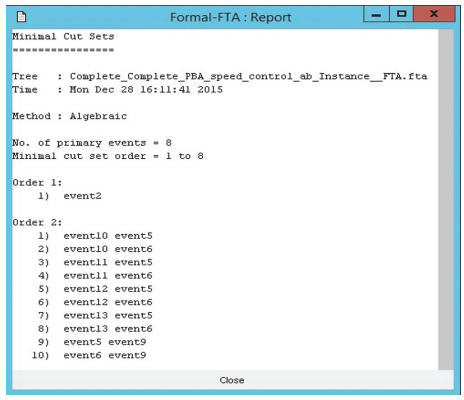

Figure 6: Minimal cut set analysis report from OpenFTA.

The FTA analysis is in conformance with MIL-STD882 standard and the generated fault tree is validated, as shown in Figure 5.

The artifacts related to FTA as specified by MIL-STD882 deal with error composites and error events. FTA is a top-down approach of analysis. The Minimal Cut Set is evaluated in the OpenFTA tool and is as shown in Figure 6.

(iii) Failure Modes and Effects Analysis (FMEA) and Functional Hazard Assessment (FHA). Analysis of the failure modes associated with the system and the determination of its effects over the hierarchical evolution, performed systematically with a bottom-up approach, is FMEA. With respect to the errors of the system, FMEA provides the information about the deficient component/models and their related effects. It also provides sufficient overview of the failing component such as its phase of failure, severity/impact, and so on. FMEA is based on the artifacts that include error propagation paths (error source, error path, and the error sink). FHA provides the possible list of error upon the synthesis of the architectural model of the system. The major artifacts from FHA comprise the source of the error and the error events, as shown in Table 1. The details of FHA are processed from the OSATE tool after the model is instantiated and the relevant error information is suitably extracted from these architecture models. The report will be in the form of an excel spreadsheet with the specification of the error event details.

5. Conclusion

In this paper, we have proposed a novel approach of safety analyses of Safety-Critical Systems using AADL and the related Error Model Annexes. In spite of the comprehensive activities involved in safety analyses, the needs for such approaches are proved to be very much necessary. This is achieved and projected with the implementation of a suitable case study, Speed Control Unit of Power-Boat Autopilot. The employment of analysis techniques such as Fault Tree Analysis (FTA), Functional Hazard Analysis (FHA), and consistency of the model along with the conduction of qualitative and quantitative reliability analyses as part of these techniques can assess the system hazards and faults. The assessment covers the generation of suitable reports justifying the analyses. These methodologies or techniques provide grant for early identification and probability of the occurrence of potential problems. This also provides a perspective to explore additional architectural properties. Reuse and analysis of the evolved models, provided with suitable extensions with limited effort, can be achieved with this approach. The overall effect induces a greater confidence over abstracted stages of development and safety analyses of these architectural models of the system. Also analysing the system based on the Safety-Critical Requirements, with the expectation of exceptional conditions, hazards are expedited in the development of Safety System architecture models which will have an impact in certifying the same. This also avoids the unnecessary certification costs by understanding the change impact or the exceptional causes impacts during system engineering.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors thank the Director of CSIR-NAL, Bengaluru, for supporting this work.

Table 1: FHA report.

Component	Error	Hazard description	Functional failure	Operational phases	Severity	Likelihood	Comment
speed_sensor	"Failure on Failure"	"Faulty speed values"	"Loss of sensor readings"	"Acquire"	Critical	Probable	"Speed values are read as faulty"
speed_sensor	"Failed on Failed"	"Failure of sensor"	"Sensor failed"	"Acquire"	Catastrophic	Frequent	"Is a major hazard. Pilot cannot estimate the speed due to sensor failure"
throttle	"Failure on Failure"	"No command inputs due to actuator failure"	"Faulty or no commands"	"Output"	Critical	Remote	"Becomes a major hazard if there are command inputs to the actuator"
throttle	"Failed on Failed"	"Faulty actuator"	"Actuator in failure state"	"Output"	Catastrophic	Frequent	"Is a major hazard. Pilot cannot control the PowerBoat with proper throttle"
interface_unit	"Failure on Failure"	"Faulty or no input values and commands"	"Loss of actuator input values"	"Input"	Critical	Probable	"Becomes a major hazard if there happens to be faulty input values"
interface_unit	"Failed on Failed"	"Failure of actuator"	"Actuator in failure state"	"Input"	Catastrophic	Frequent	"Is a major hazard. Pilot cannot set proper speed value or input commands"
display_unit_inter	"Failure on Failure"	"Improper display due to faulty values or commands"	"Faulty values or commands on display"	"Output"	Marginal	Remote	"Remote possibility with display showing faulty values or commands"
display_unit_inter	"Failed on Failed"	"Display unit not working properly"	"Faulty display unit"	"Output"	Marginal	Remote	"Not a major hazard"

References

[1] P. H. Feiler and D. P. Gluch, *Model-Based Engineering with AADL-An Introduction to the SAE Architecture Analysis & Design Language*, Pearson Education-Addison Wesley, Upper Saddle River, NJ, USA, 2012.

[2] MIL-STD882(E): Department of Defence Standard Practice, System Safety, May 2012.

[3] SAE International, "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipments," Tech. Rep. ARP-4761, 1996, http://standards.sae.org/arp4761/.

[4] RTCA, "Software Considerations in Airborne Systems and Equipment Certification," December 2011, http://www.rtca org/.

[5] A. Joshi, S. Vestal, and P. Binns, "Automatic generation of static fault trees from AADL models," in *Proceedings of the IEEE/IFIP Conference on Dependable Systems and Networks' Workshop on Dependable Systems*, Edinburgh, UK, 2007.

[6] J. Delange, *Safety Evaluation with AADLv2*, Software Engineering Institute, Carnegie Mellon University, 2013.

[7] B. Hall, K. R. Driscoll, and G. Madl, *Investigating System Dependability Modeling Using AADL*, NASA/CR-2013-217961, Honeywell International, Golden Valley, Minn, USA, 2013.

[8] Q. Li, Z. Gao, and X. Luo, "Error modeling and reliability analysis of airborne distributed software based on AADL," *Advanced Science Letters*, vol. 7, pp. 421–425, 2012.

[9] J. Delange and P. Feiler, "Architecture fault modeling with the AADL error-model annex," in *Proceedings of the 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA '14)*, pp. 361–368, Verona, Italy, August 2014.

[10] D. Powell, "Failure mode assumptions and assumption coverage," in *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing*, FTCS 22, pp. 386–395, IEEEXplore, Boston, Mass, USA, July 1992.

[11] C. J. Walter and N. Suri, "The customizable fault/error model for dependable distributed systems," *Theoretical Computer Science*, vol. 290, no. 2, pp. 1223–1251, 2003.

[12] J. Cabot and M. Gogolla, *Object Constraint Language(OCL): A Definitive Guide*, Springer, Berlin, Germany, 2010.

[13] M. Benammar and F. Belala, "How to make AADL specification more precise," *International Journal of Computer Applications*, vol. 8, no. 10, pp. 16–23, 2010.

[14] OSATE, https://wiki.sei.cmu.edu/aadl/index.php/Osate_2.

[15] OpenFTA, http://www.openfta.com/.

[16] SAE International, *Annex Behavior Language Compliance & Application Program Interface*, SAE International, Warrendale, Pa, USA, 2007.

[17] C. Li, H. Yang, and H. Liu, "An approach to modelling and analysing reliability of breeze/ADL-based software architecture," *International Journal of Automation and Computing*, In press.

[18] J. Xiang, K. Yanoo, Y. Maeno, and K. Tadano, "Automatic synthesis of static fault trees from system models," in *Proceedings of the 5th International Conference on Secure Software Integration and Reliability Improvement (SSIRI '11)*, pp. 127–136, June 2011.